作业五

1. 在一个页面大小为 1KB 的分页系统中，某进程在某时刻的页表如下，表中的所有数值均为十进制。



请将下列用十进制表示的虚地址转换为物理地址，结果用十进制表示。

(1)1052；

(2)2221；

(3)5499

1. 在可变分区存储管理下，按地址排列的内存空闲区为：W1(10K)、W2(4K)、W3(20K)、W4(18K)、W5(7K)、W6(9K)、W7(12K)和W8(15K)。对于下列的连续存储区的请求：J1(12K)、J2(10K)、J3(9K)，分别使用最先适应分配算法、最优分配算法和最差适应分配算法分配空闲区时，请给出分配过程。

在可变分区存储管理下，针对给定的内存空闲区和连续存储区请求，我们可以根据最先适应分配算法、最优分配算法和最差适应分配算法来分配空闲区。以下是各种算法的分配过程：

**最先适应分配算法（First Fit）**

1. **空闲区排列**：W1(10K)、W2(4K)、W3(20K)、W4(18K)、W5(7K)、W6(9K)、W7(12K)、W8(15K)。
2. **分配过程**：
   * **J1(12K)**：从前往后查找，W1(10K) 不足，继续查找，W2(4K) 也不足，继续到 W3(20K) 满足需求，分配给 J1，剩余 8K。此时空闲区变为：W1(10K)、W2(4K)、W3\_remain(8K)、W4(18K)、W5(7K)、W6(9K)、W7(12K)、W8(15K)。
   * **J2(10K)**：继续从前往后查找，W1(10K) 正好满足，分配给 J2。空闲区变为：W2(4K)、W3\_remain(8K)、W4(18K)、W5(7K)、W6(9K)、W7(12K)、W8(15K)。
   * **J3(9K)**：继续查找，W2(4K) 不足，W3\_remain(8K) 也不足，W4(18K) 满足需求，分配给 J3，剩余 9K。空闲区变为：W2(4K)、W3\_remain(8K)、W4\_remain(9K)、W5(7K)、W6(9K)、W7(12K)、W8(15K)。

**最优分配算法（Best Fit）**

1. **空闲区排列**（按大小升序排列）：W2(4K)、W5(7K)、W1(10K)、W6(9K)、W7(12K)、W8(15K)、W3(20K)、W4(18K)。
2. **分配过程**：
   * **J1(12K)**：从小到大查找，直到找到 W7(12K) 满足需求，分配给 J1，无剩余。空闲区变为：W2(4K)、W5(7K)、W1(10K)、W6(9K)、W8(15K)、W3(20K)、W4(18K)。
   * **J2(10K)**：继续查找，W1(10K) 满足需求，分配给 J2，无剩余。空闲区变为：W2(4K)、W5(7K)、W6(9K)、W8(15K)、W3(20K)、W4(18K)。
   * **J3(9K)**：继续查找，W6(9K) 满足需求，分配给 J3，无剩余。空闲区变为：W2(4K)、W5(7K)、W8(15K)、W3(20K)、W4(18K)。

**最差适应分配算法（Worst Fit）**

1. **空闲区排列**（按大小降序排列）：W3(20K)、W4(18K)、W8(15K)、W7(12K)、W1(10K)、W6(9K)、W5(7K)、W2(4K)。
2. **分配过程**：
   * **J1(12K)**：从大到小查找，W3(20K) 满足需求，分配给 J1，剩余 8K。空闲区变为：W3\_remain(8K)、W4(18K)、W8(15K)、W7(12K)、W1(10K)、W6(9K)、W5(7K)、W2(4K)。
   * **J2(10K)**：继续查找，W4(18K) 满足需求，分配给 J2，剩余 8K。空闲区变为：W3\_remain(8K)、W4\_remain(8K)、W8(15K)、W7(12K)、W1(10K)、W6(9K)、W5(7K)、W2(4K)。
   * **J3(9K)**：继续查找，W8(15K) 满足需求，分配给 J3，剩余 6K。空闲区变为：W3\_remain(8K)、W4\_remain(8K)、W8\_remain(6K)、W7(12K)、W1(10K)、W6(9K)、W5(7K)、W2(4K)。

这样，我们根据最先适应、最优分配和最差适应三种算法，对给定的空闲区和作业请求进行了分配。

1. 有一个具有40个磁道的盘面，编号为0～39，当前磁头位于第11道，且由内往外运动。现有如下柱面访问请求序列：1，36，16，34，9，12；若分别采用最短寻道时间优先算法和扫描算法进行驱动调度，请分别给出处理请求的顺序，并计算磁头移动的总磁道数。

首先，我们来理解题目中的两种磁盘调度算法：

1. **最短寻道时间优先（SSTF）算法**：该算法选择离当前磁头位置最近的请求进行处理，以最小化磁头移动。
2. **扫描（SCAN）算法**：磁头从里向外移动，直到处理完所有外道的请求，然后磁头返回最里道，再开始新一轮的扫描。在处理请求时，磁头只向一个方向移动，直到该方向没有请求为止，然后改变方向。

现在，我们根据这两种算法来处理请求序列：1，36，16，34，9，12，并计算磁头移动的总磁道数。

**最短寻道时间优先（SSTF）算法**

1. **当前磁头位置**：11
2. **请求序列**：1，36，16，34，9，12
3. **处理顺序**：
   * 从11开始，离11最近的请求是12（移动1道）
   * 然后是9（从12向内移动3道）
   * 接着是1（从9向内移动8道）
   * 然后是16（从1向外移动15道）
   * 接着是34（从16向外移动18道）
   * 最后是36（从34向外移动2道）
4. **总磁道数**：1 + 3 + 8 + 15 + 18 + 2 = 47道

**扫描（SCAN）算法**

1. **当前磁头位置**：11，且由内往外运动
2. **请求序列**：1，36，16，34，9，12
3. **处理顺序**：
   * 磁头正在向外移动，所以先处理所有外道的请求
   * 首先是12（从11向外移动1道）
   * 然后是16（从12向外移动4道）
   * 接着是34（从16向外移动18道）
   * 最后是36（从34向外移动2道）
   * 此时磁头到达最外道，开始反向移动，处理内道的请求
   * 首先是9（从36向内移动27道）
   * 接着是1（从9向内移动8道）
   * 注意：虽然SCAN算法通常会在到达一端后反向移动到另一端，但在这个特定问题中，我们只需要计算磁头移动的总磁道数，所以不需要模拟完整的扫描过程。
4. **总磁道数**：1 + 4 + 18 + 2 + 27 + 8 = 60道

所以，按照最短寻道时间优先（SSTF）算法，磁头移动的总磁道数是47道；按照扫描（SCAN）算法，磁头移动的总磁道数是60道。